SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1549

Određivanje vlastitog gibanja kamere primjenom naučene korespondencijske metrike

Dario Smolčić

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 15. ožujka 2017.

DIPLOMSKI ZADATAK br. 1549

Pristupnik: **Dario Smolčić (0036473172)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Određivanje vlastitog gibanja kamere primjenom naučene korespondencijske metrike**

Opis zadatka:

Određivanje gibanja kamere analizom slijeda stereoskopskih slika važno je područje primjene računalnog vida. Većina pristupa za rješavanje tog problema temelji se na korespondenciji točkastih značajki u četvorkama slika pribavljenih u susjednim vremenskim trenutcima. Tema rada je istražiti mogućnost ostvarivanja korespondencije ugrađivanjem slikovnih okana u visokodimenzionalni metrički prostor.
U okviru rada, potrebno je proučiti postupke praćenja značajki te procjene gibanja iz literature. Evaluirati preciznost tih postupaka na georeferenciranim sljedovima slika. Preuzeti i vrednovati parametre ugrađivanja. Razviti i integrirati poboljšane module za praćenje značajki te optimiranje gibanja kamere. Uhodati postupke validiranja hiperparametara. Primijeniti razvijene postupke na georeferenciranim sljedovima slika. Prikazati i ocijeniti ostvarene rezultate. Predložiti pravce budućeg razvoja.
Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne sljedove slika i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 29. lipnja 2017.

Mentor:

Izv. prof. dr. sc. Siniša Šegvić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Određivanje vlastitog gibanja kamere je važan zadatak u području računalnog vida i robotike. Ovaj postupak je preduvjet za brojne primjene poput detekcije prepreka, autonomne vožnje, sustava za proširenu stvarnost i simultane lokalizacije i mapiranja (SLAM). To je postupak određivanja trajektorije kamere samo na temelju snimke s kamere. Određuje se svih 6 stupnjeva slobode kamere tj. njezina translacija i rotacija. Za ovaj postupak se u literaturi često koristi naziv vizualna odometrija [10]. Gibanje kamere se određuje inkrementalno promatrajući promjene slika na kameri inducirane samim gibanjem.

Određivanje vlastitog gibanja kamere osigurava procjene gibanja vozila (agenta) s greškama procjene u rasponu od 0.1 do 2% [12]. Ovo svojstvo čini ovaj postupak odličnom nadopunom različitim navigacijskim sustavima poput globalnog sustava pozicioniranja (GPS), jedinica za mjerenje inercije (IMU), odometriji kotača i laserskoj odometriji. Također, određivanje vlastitog gibanja kamere ima veliku primjenu u područjima gdje su klasične metode navigacije nedostupne, npr. pod vodom ili u zraku (nema GPS-a) a i NASA koristi ovaj postupak za navigaciju rovera na Marsu.

Ovaj rad opisuje sustav za određivanje gibanja kamere analizom slijeda stereoskopskih slika. Većina pristupa za rješavanje tog problema temelji se na korespondenciji točkastih značajki u četvorkama slika pribavljenih u susjednim vremenskim trenutcima. U ovom radu je istražena mogućnost ostvarivanja korespondencije ugrađivanjem slikovnih okana u visokodimenzionalni metrički prostor. Kao referentni sustav je korištena biblioteka libviso2 [4] za određivanje vlastitog gibanja kamere koja je nadograđivana za specifične postupke. Estimacija gibanja je nadograđena tako da se rotacija i translacija estimiraju odvojeno. Ideja za ovu nadogradu je preuzeta iz [3]. Korespondencijska metrika je naučena korištenjem konvolucijskih neuronskih mreža. Metrika je naučena na stereo parovima rektificiranih slika s ciljem točnijeg određivanja dispariteta. Cijeli postupak je opisan u [15]. Ovako naučena korespondencijska metrika postiže točnost od 85% u postupku određivanja dispariteta točaka. U ovom radu je preuzeta ta ista metrika (koristi se gotovi naučeni model) te se koristi za odre-

đivanje korespondencija između točkastih značajki. Napravljena je evaluacija metrike te je analiziran njezin utjecaj na točnost estimacije gibanja u različitim uvjetima.

U drugom poglavlju su opisani postupci detekcije značajki i određivanja njihovih korespondencija. Tu je opisana naučena metrika koja je glavni fokus ovog rada. Nakon toga u trećem poglavlju je opisan cijeli sustav za određivanje vlastitog gibanja kamere. U četvrtom poglavlju je opisana programska implementacija i ispitni skupovi. U petom poglavlju se nalazi evaluacija metrike i rezultati estimacije gibanja s različitim parametrima. Konačno, zaključak daje buduće mogućnosti rada na problemu i moguća poboljšanja.

2. Detekcija značajki i određivanje korespondencija

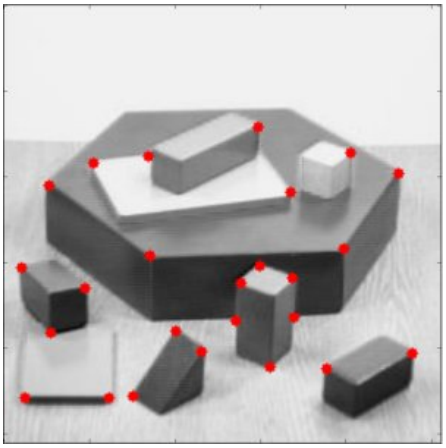
U ovom poglavlju ćemo opisati postupke detekcije značajki i određivanja njihovih korespondencija. Postoje dva glavna pristupa detekciji značajki i njihovih korespondencija. Prvi je pronalazak značajki u jednom paru slika (lijeva i desna) i njihovo praćenje u narednim slikama koristeći lokalne tehnike pretraživanja. Drugi pristup je neovisna detekcija značajki na svim slikama te pronalazak njihovih korespondencija temeljem neke mjere sličnosti između njihovih deskriptora. U ovom radu je korišten drugi postupak.

2.1. Detekcija značajki

Tijekom ovog koraka na slikama se traže istaknute točke koje će se vrlo vjerojatno detektirati i na drugim slikama. Lokalna značajka je uzorak na slici koji se razlikuje od svojih susjeda u intenzitetu, boji i teksturi. Za određivanje vlastitog gibanja kamere su važni detektori točaka poput kuteva i mrlja zato što se njihova pozicija u slici može precizno odrediti. Primjer detektiranih kuteva i mrlja možemo vidjeti na slici 2.1.

Detektiranje kuteva je računski efikasnije naspram detekcije mrlja ali zato detekcija mrlja daje karakterističnije značajke. Također kutevi se lakše lokaliziraju u poziciji slike dok se kod mrlja lakše odredi njihova veličina.

Poželjna svojstva dobrog detektora značajki su: točnost lokalizacije (pozicije i veličine značajke), ponovljivost (veliki broj značajki bi trebao biti ponovno detektiran u sljedećim slikama), računalna efikasnost, robusnost (na šum, zamućivanje i slično), karakterističnost (detektirane značajke su specifičnog izgleda) i invarijantnost na fotometrijske promjene (npr. osvjetljenje) i geometrijske promjene (rotacija, skaliranje, perspektivna distorzija). Postoje brojni detektori značajki. Najpoznatiji detektori kuteva su: Harris [7], Shi-Tomasi [13] i FAST [11]. Najpoznatiji detektori mrlja su: SIFT [8], SURF [2] i CENSURE [1]. Svaki detektor značajki ima svoje prednosti i mane



(a) Kutevi

(b) Mrlje

Slika 2.1: Detekcija značajki

naspram drugih detektora pa zato izbor detektora najviše ovisi o specifičnoj primjeni.

Svaki detektor značajki sadrži dva koraka. Prvi korak je primjena neke funkcije na cijelu sliku, npr. razlika Gaussiana se primjenjuje kod SIFT detektora. Drugi korak je detekcija svih lokalnih minimuma ili maksimuma izlaza prethodno primijenjene funkcije. Detektirani lokalni minimumi i maksimumi predstavljaju značajke. Kako bi detektor bio invarijantan na skaliranje može se primijeniti na uvećanim i umanjenim verzijama iste slike. Invarijantnost na perspektivne promjene se postiže aproksimacijom perspektivne distorzije afinom funkcijom.

Slijede kratki opisi tri različita detektora korištena u ovom radu: detektor iz libviso2 biblioteke, Harris i FAST.

2.1.1. Libviso detektor

Kako bi se pronašle lokacije stabilnih značajki, najprije se ulazne slike filtriraju s maskama dimenzija 5×5 za detekciju kuteva i mrlja. Korištene maske prikazane su na slici 2.3. Sljedeći korak je potiskivanje odziva koji nisu maksimalni ni minimalni (engl. non-maximum and non-minimum-suppression), čime preostale značajke pripadaju jednoj od četiri skupine:

1. mrlja, maksimum
2. mrlja, minimum
3. kut, maksimum
4. kut, minimum

-1	-1	0	+1	+1
-1	-1	0	+1	+1
0	0	0	0	0
+1	+1	0	-1	-1
+1	+1	0	-1	-1

(a) Kutevi

-1	-1	-1	-1	-1
-1	+1	+1	+1	-1
-1	+1	+8	+1	-1
-1	+1	+1	+1	-1
-1	-1	-1	-1	-1

(b) Mrlje

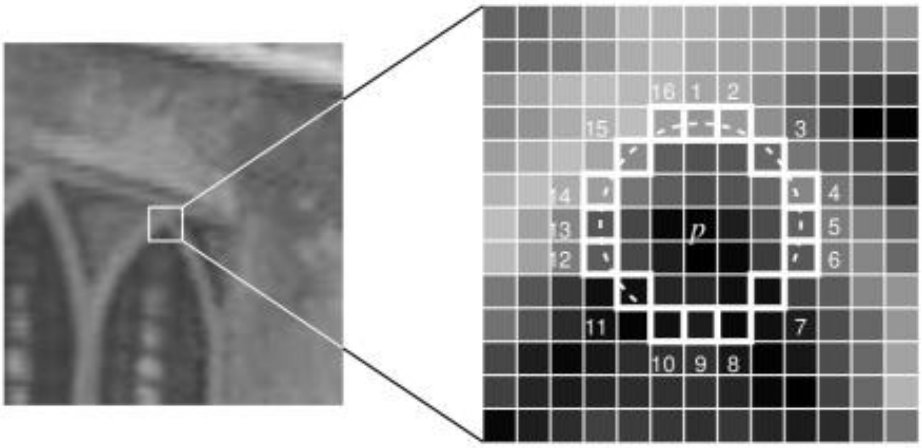
Slika 2.2: Maske za detekciju značajki

Korespondencije se određuju samo unutar istih skupina značajki.

2.1.2. FAST detektor

FAST detektor je detektor kuteva detaljno opisan u [11]. Za svaki piksel p u slici za koji se želi provjeriti da li je kut ili ne se radi sljedeće:

- Uzima se intenzitet piksela I_p.
- Određuje se prikladna vrijednost praga t.
- Promatra se 16 piksela u krugu oko piksela p.
- Promatrani piksel je kut ako postoji skup od n spojenih piksela u krugu od 16 piksela koji su ili svjetliji od $I_p + t$ ili tamniji od $I_p - t$.



Slika 2.3: Prikaz ispitivanja piksela za FAST značajke

Kako bi se postupak ubrzao koristi se naučeno stablo odluke za određivanje da li je piksel kut ili ne.

2.1.3. Harrisov detektor

Harrisov detektor [6] je također detektor kutova. Kutevi su definirani kao područja u slici s velikom varijacijom u intenzitetu u svim smjerovima. Harrisov detektor koristi upravo ovu činjenicu u matematičkoj formi. On traži razlike u intenzitetu prilikom pomicanja u svim smjerovima (u, v) od promatranog piksela:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{2.1}$$

$I(x, y)$ označava intenzitet slike na koordinatama x i y, a $w(x, y)$ je funkcija koja označava koje piksele želimo promatrati, tj. pravokutnu veličinu prozora oko potencijalne značajke.

Ako su razlike u intenzitetu prilikom pomicanja u svim smjerovima velike onda je pronađen kut. U stvarnosti se ova funkcija ne računa izravno nego nas samo zanima njezin oblik u području $(u, v) = (0, 0)$ koji se dobiva iz Taylorove ekspanzije funkcije oko točke $(0, 0)$. Detaljniji opis cijelog postupka se nalazi u [6].

2.2. Određivanje korespondencija

U ovom koraku se traže korespondentne značajke u drugim slikama. Najjednostavniji način pronalaženja korespondencija između dvije slike jest usporedba svih deskriptora značajki prve slike sa svim deskriptorima značajki druge slike. Najbolja korespondencija značajke se odabire kao ona značajka na drugoj slici čiji je deskriptor najsličniji traženoj značajki. U ovom slučaju se može dogoditi da ista značajka na drugoj slici upari s više značajki na prvoj slici (očito samo jedan par može biti točan). Za odluku koje korespondencije prihvatiti koristi se dodatno provjera unakrsne konzistentnosti (engl. mutual consistency check) koja rješava prethodno spomenuti problem. Time se ujedno i značajke druge slike uparuju sa značajkama prve slike. To znači da svaka značajka na prvoj slici ima preferirani par na drugoj slici, a ujedno i svaka značajka na drugoj slici ima preferirani par na prvoj slici. Samo one značajke koje se međusobno imaju kao preferirane parove se odabiru kao ispravne korespondencije. Na slici 2.4 su prikazane korespondencije detektiranih značajki na dvije različite slike.

Mana ovakvog pristupa određivanju korespondencija jest to što je postupak kvadratne složenosti s obzirom na broj značajki što može postati nepraktično ako je broj značajki velik (nekoliko tisuća). Bolji pristup je traženje potencijalnih korespondencija u područjima druge slike gdje bi se najviše očekivalo da se značajka pojavi. Dodatno u stereoskopskim sustavima se može značajno ubrzati traženje korespondencija između

Slika 2.4: Korespondencije značajki

lijeve i desne slike rektifikacijom slika. Rektifikacijom se slike postavljaju u koplanarnu ravninu i dijele os x što znači da će sve korespondencije biti na istim horizontalnim linijama u obje slike. Time se prostor pretraživanja smanji samo na značajke koje se nalaze na istoj liniji. Rektifikacija se može efikasno izvesti na grafičkim karticama.

Alternativa neovisnom pronalasku značajki u svakoj slici te određivanju korespondencija između takvih značajki je da se značajke detektiraju u prvoj slici i onda se traže njihove korespondencije u sljedećim slikama. Značajke u sljedećim slikama se ne traže neovisno nego se pronalaze najbolje korespondencije s obzirom na početne detektirane značajke. Ovakav pristup se često koristi kod traženja korespondencija između lijeve i desne slike kod rektificiranih slika. Značajke se detektiraju u lijevoj slici s nekim detektorom a u desnoj slici se za svaku detektiranu značajku u lijevoj slici traži njezina korespondencija na istoj horizontalnoj liniji. Detektor značajki se koristi samo na lijevoj slici dok se na desnoj slici traže točke koje su najsličnije značajkama detektiranima na lijevoj slici. Pošto su slike rektificirane pretražuju se samo iste linije.

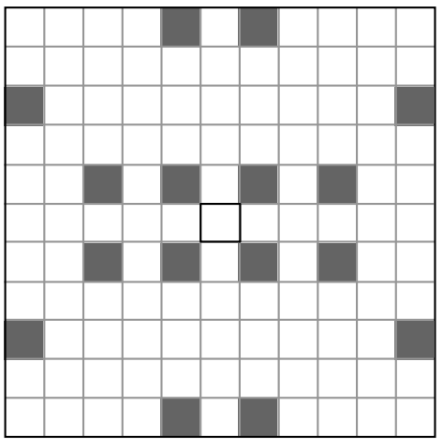
2.3. Deskriptori značajki i korespondencijske metrike

Regije oko svake detektirane značajke se spremaju u kompaktne deskriptore koji se mogu uspoređivati između značajki. Najjednostavniji deskriptor značajke jest njezin izgled tj. intenzitet piksela oko značajke. U tom slučaju metrike poput sume kvadrata razlike (SSD) ili normalizirana unakrsna korelacija (NCC) se mogu koristiti za uspoređivanje značajki. Robusnija mjera sličnosti jest Census transformacija koja pretvara okvir oko značajke u binarni vektor koji predstavlja koji pikseli imaju veće ili manje intenzitete od centralnog piksela. Sličnost se onda mjeri Hammingovom udaljenošću.

U ovom radu su se koristile dvije vrste deskriptora i korespondencijskih metrika.

2.3.1. Libviso deskriptor i SAD metrika

U ovom potpoglavlju opisujemo standardne deskriptore i metriku iz libviso2 biblioteke. Deskriptori libviso značajki se dobivaju tako da se ulazne slike konvoluiraju Sobelovim filtrom te se oko značajki uzimaju područja veličine 11×11 piksela. Radi ubrzanja postupka, odziv Sobelovog filtra se kvantizira na 8 bitova te se razmatra samo 16 lokacija odziva, kao što prikazuje slika 2.5.

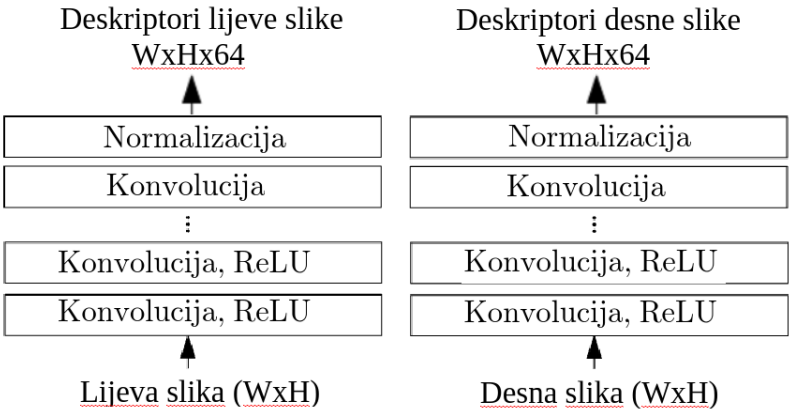


Slika 2.5: Libviso deskriptor za usporedbu značajki

Kao metrika za usporedbu ovih deskriptora se koristi suma apsolutnih udaljenosti (SAD). Znači da se za vektor od 16 elemenata zbrajaju apsolutne razlike svakog elementa. Što je ta suma manja to su značajke sličnije.

2.3.2. Konvolucijski deskriptori i metrika

Za opisivanje značajki se koriste konvolucijski deskriptori koji se dobiju kao izlaz iz naučene konvolucijske neuronske mreže. Konvolucijska mreža (slika 2.6) se sastoji od dvije podmreže, (za značajke iz dvije slike koje se uspoređuju). Svaka podmreža se sastoji od niza konvolucijskih slojeva iza kojih se nalaze linearne rektificirane jedinice (ReLU). Ulaz u svaku podmrežu je jedna slika dimenzija $W \times H$, a izlaz je blok dimenzija $W \times H \times 64$. To znači da je svaki piksel ulazne slike opisan s vektorom od 64 elementa. Ti vektori su normalizirani. S jednim unaprijednim prolazom kroz mrežu možemo dobiti deskriptore za svaki piksel u obje slike. Detaljniji opis konvolucijske neuronske mreže, postupka učenja i parametara se nalazi u [15].



Slika 2.6: Arhitektura konvolucijske mreže

Ove konvolucijske deskriptore uspoređujemo tako da uzmemo njihov skalarni produkt. Pošto su deskriptori normalizirani njihov skalarni produkt je kosinusna sličnost (formula 2.2) između vektora koja može biti u rasponu [-1, 1], gdje bi -1, značio da su značajke apsolutno suprotne, tj. nisu nimalo slične, a 1 da su značajke identične.

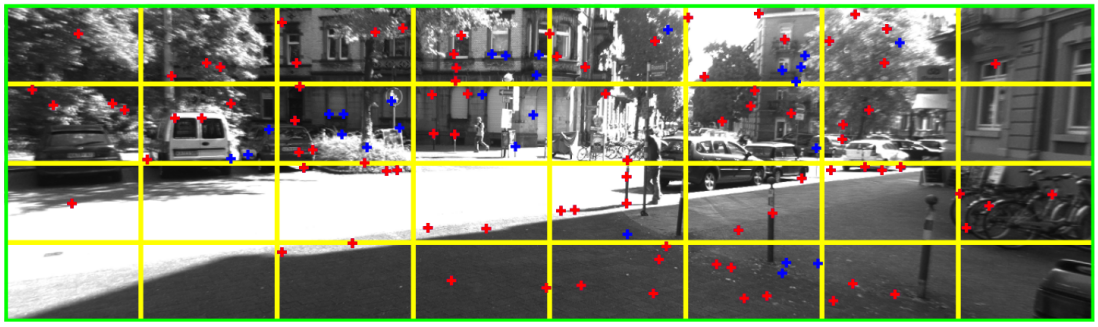
$$s = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{64} A_i B_i \tag{2.2}$$

2.4. Grupiranje značajki

Precizno određivanje gibanja kamere pretpostavlja da se u estimaciji koriste značajke koje su blizu i daleko od kamere i da su značajke jednoliko raspoređene po slici. Ovi uvjeti se mogu ispuniti metodom grupiranja (engl. bucketing). Slika se dijeli na područja veličine 50×50 piksela. U svakom tom području se zadržava ograničeni broj značajki. Prilikom odabira značajki u svakom području značajke se sortiraju prema određenom kriteriju te se uzimaju najbolje značajke. Kriteriji sortiranja mogu biti sljedeći:

- Starost značajke - pouzdanije su značajke koje su se pojavljivale u većem broju prethodnih slika.
- Pogreška ili sličnost korespondencijske metrike.
- Jačina odziva prilikom detekcije značajke.

Na slici 2.7 se vidi primjer grupiranja značajki gdje se unutar svakog područja zadržava ukupno 4 značajke. S crvenom bojom su označene zadržane značajke a s plavom odbačene.



Slika 2.7: Grupiranje značajki

3. Određivanje gibanja kamere

3.1. Formulacija problema

Agent se kreće kroz okolinu s pričvršćenim stereo kamerama koje dohvaćaju slike okoline u diskretnim vremenskim trenutcima k. Za svaki vremenski trenutak postoji lijeva i desna slika koje ćemo označavati s $I_{l,k}$ i $I_{r,k}$ gdje je l lijevo (engl. left), r desno (engl. right) a k indeks vremenskog trenutka. Bez gubitka generalizacije koordinatni sustav lijeve kamere će se smatrati izvornim koordinatnim sustavom agenta. Položaji kamere u dva vremenski susjedna trenutka $k-1$ i k su povezana transformacijom $T_{k,k-1} \in \mathbb{R}^{4\times4}$ sljedećeg oblika:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

Gdje je $R_{k,k-1} \in \mathbb{R}^{3\times3}$ matrica rotacije a $t_{k,k-1} \in \mathbb{R}^{3\times1}$ vektor translacije.

Označimo s C_n trenutni položaj kamere u trenutku n koji sadrži i orijentaciju i translaciju kamere u svjetskim koordinatama. C_n je istog oblika kao već prikazana matrica $T_{k,k-1}$. Onda vrijedi sljedeća rekurzivna formula:

$$C_n = C_{n-1}T_{n,n-1} \tag{3.2}$$

Trenutni položaj kamere C_n možemo dobiti primjenjujući sve transformacije $T_{k,k-1}$ $(k = 1 \dots n)$ krenuvši od početnog položaja. C_0 je položaj kamere u početnom trenutku koji se može zadati proizvoljno.

Glavni zadatak određivanja vlastitog gibanja kamere je računanje transformacije $T_{k,k-1}$ na temelju slika $I_{l,k-1}$, $I_{r,k-1}$, $I_{l,k}$ i $I_{r,k}$ te određivanje potpune trajektorije kamere tj. određivanje položaja kamere u svakom vremenskom trenutku $C_{0:n} = \{C_0, ..., C_n\}$. Postoje dva glavna pristupa ovom problemu: metode temeljene na izgledu (globalne metode) koje koriste intenzitet svakog piksela u slici i metode temeljene na značajkama koje koriste istaknute značajke na slikama koje se detektiraju i prate. U ovom radu će biti opisane samo metode temeljene na značajkama.

3.2. Model kamere

Koristit ćemo točkasti model kamere s perspektivnom projekcijom π. Neka je $X = [x, y, z]^T$ točka u sceni prikazana u svjetskim koordinatama a $q = (u, v)^T$ njezina projekcija na ravninu slike. Projekcija π projicira točku X u točku slike q prema sljedećoj formuli:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \pi(X, R, t) = \begin{pmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{pmatrix} [R|t] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.3}$$

Prvi dio je matrica kamere s intrinzičnim parametrima kamere f, c_u i c_v. Točka (c_u, c_v) je točka u kojoj os z siječe ravninu slike. S $[R|t]$ je označena ekstrinsična matrica kamere sljedećeg oblika:

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{3.4}$$

Primijetimo da $[R|t]$ u sebi ne sadrži rotaciju i translaciju kamere u svjetskim koordinatama nego ona opisuje kako transformirati točke iz svjetskih koordinata u koordinatni sustav kamere.

3.3. Funkcije pogreške

Označimo točku slike s $q = (u, v)^T$. Sada definirajmo točke $q^k_{i,t-1}$ u prethodnom trenutku $(t-1)$ i $q^k_{i,t}$ u trenutnom trenutku (t) gdje je $i \in [1, N]$ indeks točke a $k \in \{l, r\}$ oznaka kojom označavamo da li se točka nalazi na lijevoj ili desnoj slici. Označimo također s $X_{i,t}$ 3D točku u trenutnom trenutku (t):

$$X_{i,t} = (x, y, z)^\top = t(q^l_{i,t}, q^r_{i,t}) \tag{3.5}$$

gdje je $t()$ funkcija koja iz lijeve i desne točke na slikama triangulira 3D točku u svjetskom koordinatnom sustavu. Sada možemo formulirati funkciju pogreške na dva načina. Generalno rješenje se sastoji od pronalaženja matrice rotacije R i vektora translacije t koji minimiziraju L2 udaljenost između 2 skupa 3D točki: točke u trenutnom trenutku (t) i točke u prethodnom trenutku $(t-1)$.

$$\underset{T}{\operatorname{argmin}} \sum_i \|\tilde{X}_{i,t} - T\tilde{X}_{i,t-1}\| \tag{3.6}$$

Gdje su $\tilde{X}_{i,t-1}$ i $\tilde{X}_{i,t}$ homogene koordinate 3D točaka a T transformacijska matrica koja opisuje gibanje kamere istog oblika kao u jednadžbi 3.1. Važno je spomenuti da točke a istim indeksom i odgovaraju istim značajkama u uzastopnim slikama tj. predstavljaju poziciju istog objekta u stvarnom svijetu. Određivanje korespondentnih značajki je obrađeno u poglavlju 2.

Umjesto prethodne funkcije pogreške danas se koristi reprojekcijska pogreška koja je prvi put predložena u [10] i koja se pokazala puno efikasnijom:

$$\underset{R,t}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{k}^{\{l,r\}} \|q_{i,t}^{k} - \pi(X_{i,t-1}, R, t_k)\| \tag{3.7}$$

$$t_l = t, \qquad t_r = t_l - (b, 0, 0)^\top \tag{3.8}$$

gdje je π perspektivna projekcija prikazana jednadžbom 3.3. Jednadžba 3.7 pokazuje da se traži transformacija koja minimizira udaljenost između točaka na slikama dobivenih u trenutnom trenutku (t) i re-projekcija trianguliranih 3D točaka iz prethodnog trenutka $(t-1)$. Reprojekcijska pogreška se pokazala kao puno bolji izbor za određivanje vlastitog gibanja kamere od pogreške iz jednadžbe 3.6. Razlog tome je nesigurnost dubine trianguliranih 3D točki. Uzrok te nesigurnosti jest činjenica da pomicanje 3D točke u smjeru dubine ne utječe puno na reprojekcije točaka. Kada se dvije točke trianguliraju u dva različita vremenska trenutka dolazi do velikih nekonzistentnosti u smjeru dubine što uvelike utječe na procjenu gibanja. Kod reprojekcijske pogreške se 3D točke iz prethodnog trenutka $(t-1)$ ponovno reprojiciraju u koordinate slike te se uspoređuju u 2D koordinatnom prostoru slike. Time je uvelike uklonjena nesigurnost triangulacije.

Napomenimo da navedena formula reprojekcijske pogreške vrijedi ako je stereo sustav kalibriran i ako su slike rektificirane. Primijetimo da se optimizira šest parametara: tri koja opisuju rotacijsku matricu R i tri koja opisuju translaciju t. Jednadžba 3.8 opisuje odnos između translacije lijeve i desne kamere u slučaju kada su slike rektificirane. Rotacijska matrica R je ista i za lijevu i za desnu kameru jer su kamere nakon rektifikacije poravnate tako da su im ravnine slika koplanarne i imaju istu os x.

Gibanje 3D točaka s obzirom na statičku kameru je određeno transformacijskom matricom $[R|t]$, a da bismo dobili gibanje kamere potrebno je napraviti inverz $[R|t]^{-1}$.

Navedene funkcije pogreške se mogu minimizirati analitičkim ili iterativnim optimizacijskim metodama (Gauss-Newtonova metoda, Newtonova metoda, dekompozicija singularnih vrijednosti) koje se neće obrađivati u ovom radu. Minimalni broj točaka za minimiziranje obje funkcije pogreške je tri.

3.4. Odvojena estimacija rotacije i translacije

Kao što je prethodno navedeno rotacija i translacija se mogu odrediti istovremeno minimizacijom reprojekcijske pogreške (formula 3.7). Međutim u ovom radu se koristi odvojena estimacija rotacije i translacije prema uzoru na [3].

3.4.1. Estimacija rotacije

Rotacija se estimira monokularnom metodom koristeći samo slike s lijeve kamere. Koristi se metoda pet točaka opisana u [9]. Metoda pet točaka se koristi u kombinaciji s RANSAC-om. Određeni broj nasumičnih podskupova od pet točaka se odabire iz ukupnog skupa točaka korespondencije. Esencijalna matrica se računa za svaki od tih podskupova. Konačno, esencijalna matrica koja ima najveći skup dobrih korespondencija (engl. inlier) među svim točkama se odabire kao konačno rješenje. Dekompozicijom matrice se dobiva rotacijska matrica. Optimiranje rješenja s više od pet točaka, npr. s odgovarajućim dodatnim dobrim korespondencijama nije izvođeno.

Glavne prednosti monokularne estimacije rotacije s metodom pet točaka su:

- Ima rješenje u zatvorenoj formi.
- Rješenje se dobiva iz minimalnog broja točaka. Vjerojatnost da je jedna od njih kriva je mala.
- Monokularna metoda ne podliježe pogreškama u kalibraciji između lijeve i desne kamere.

Rotacija se također može dobiti ujednačavanjem rotacije iz prethodnog koraka i trenutnog. Možemo dobiti rotaciju između trenutnog koraka t i koraka $t-2$ koju ćemo označiti sa q_{t-2}^{t}. Pošto je prethodna rotacija q_{t-2}^{t-1} poznata, trenutna rotacija se može alternativno izračunati prema sljedećoj formuli:

$$q_{t-1}^{t\,\prime} = (q_{t-2}^{t-1})^{-1} q_{t-2}^{t} \tag{3.9}$$

Na temelju dva različita mjerenja trenutne rotacije konačna rotacija se može dobiti postupkom sferične linearne interpolacije prema sljedećoj formuli:

$$Q_{t-1}^{t} = q_{t-1}^{t}((q_{t-1}^{t})^{-1} q_{t-1}^{t\,\prime})^{0.5} \tag{3.10}$$

Važno je istaknuti da su oznake u formulama kvaternioni a ne rotacijske matrice.

3.4.2. Estimacija translacije

Translacija se određuje minimizacijom reprojekcijske pogreške. Odrede se 3D točke iz prethodnog koraka postupkom triangulacije koristeći korespondencije između lijeve i desne slike prethodnog koraka. Te 3D točke se reprojiciraju na ravninu slike u trenutnom koraku prema formuli 3.3. Kao $[R|t]$ se postavlja matrica s rotacijom i translacijom iz prethodnog u trenutni korak. Translacija se onda računa iterativno minimizirajući reprojekcijsku pogrešku (formula 3.7). Pogreška se minimizira samo s obzirom na translaciju dok je rotacija postavljena kao ona dobivena prethodno opisanim monokularnim postupkom. Minimizacija se obavlja Gauss- Newtonovom metodom prvog reda. Za minimizaciju se koriste tri točke a najbolje rješenje se određuje korištenjem RANSAC-a.

3.5. Uklanjanje krivih korespondencija

Među detektiranim korespondencijama se uvijek nalazi određeni broj krivih korespondencija (engl. *outliers*). Mogući uzroci krivih korespondencija su šum u slici, zaklanjanje objekata, zamućeni dijelovi slike, promjene u osvjetljenju ili kutu gledanja te bilo kakve druge promjene na koje matematički model detekcije korespondencija nije otporan. Krive korespondencije uvelike utječu na određivanje vlastitog gibanja kamere te uvode velike greške u model. Zato je od velike važnosti imati dobar sustav za uklanjanje krivih korespondencija.

3.5.1. Robusna estimacija konsenzusom slučajnih uzoraka

Konsenzus slučajnih uzoraka (RANSAC) je iterativna metoda za procjenu parametara matematičkog modela iz skupa podataka te ima široku primjenu. Glavna ideja je da se slučajnim odabirom minimalnog uzorka hipotetizira traženi model. Hipoteze se zatim evaluiraju nad svim elementima ulaznog skupa podataka. Ona hipoteza koja je u najvećem konsenzusu s ulaznim skupom podataka (najbolje "paše" na podatke) se odabire kao najbolja. U području određivanja vlastitog gibanja kamere minimalni uzorak je minimalni broj korespondencija za određivanje gibanja dok je hipoteza modela izračunato relativno gibanje između dva koraka $[R|t]$. U razvijenom modelu se RANSAC koristi dva puta. Jednom za izračun rotacije R, a onda drugi put za izračun translacije t. U opisu koraka RANSAC-a ćemo koristiti notaciju $[R|t]$ radi jednostavnosti iako se u razvijenom modelu R i t estimiraju odvojeno.

Koraci RANSAC-a su sljedeći:

1. Nasumično odaberi minimalni broj korespondencija iz skupa svih korespondencija.

2. Izračunaj $[R|t]$ na temelju tih korespondencija

3. Izračunaj pogrešku (ili udaljenost) ostalih korespondencija s obzirom na model $[R|t]$.

4. Konstruiraj skup dobrih korespondencija. To su korespondencije čija je pogreška manja od nekog fiksnog praga.

5. Ponavljaj od točke 1 dok nije proveden maksimalni broj iteracija.

6. Skup s najvećim brojem dobrih korespondencija je odabran kao rješenje. Za ovaj skup pretpostavljamo da ne sadrži krive korespondencije.

7. Izračunaj model $[R|t]$ koristeći sve korespondencije iz odabranog skupa.

Primijetimo da je RANSAC probabilistička metoda te je nedeterministička s obzirom na različita pokretanja postupka. Ipak rješenja postaju stabilna s većim brojem iteracija. Broj potrebnih iteracija kako bi se garantirala ispravnost rješenja se može izračunati prema sljedećoj formuli:

$$n = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{3.11}$$

Gdje je s minimalan broj korespondencija iz kojih se može izračunati model $[R|t]$, ϵ pretpostavljeni postotak krivih korespondencija u skupu svih korespondencija i p vjerojatnost da barem jedan od slučajnih odabira bude skup koji ne sadrži krive korespondencije.

3.6. Algoritam određivanja vlastitog gibanja kamere

Algoritmi za određivanje vlastitog gibanja kamere mogu se podijeliti na algoritme koji koriste uparivanje značajki između susjednih slika te algoritme koji prate značajke kroz slijed slika. U ovom radu se koristi klasični sustav određivanja vlastitog gibanja kamere koji koristi uparivanje značajki između susjednih slika. Algoritam je podijeljen u sljedeće korake:

1. Dohvati lijevu i desnu sliku I_l, I_r.

2. Odredi korespondentne značajke između trenutne i prethodne lijeve slike.

3. Ukloni krive korespondencije (engl. outliere).
4. Odredi gibanje kamere na temelju 3D točaka iz prethodne iteracije i 2D točaka trenutne iteracije minimizirajući reprojekcijsku pogrešku.
5. Odredi korespondentne značajke između trenutne lijeve i desne slike.
6. Trianguliraj značajke u 3D točke (za korištenje u idućem koraku).
7. Ponavljaj od točke 1.

Primijetimo da je ovakav proces sklon akumuliranju greške s vremenom. Greške u trenutnim koracima se propagiraju u iduće korake te greška raste iterativno. Najveće greške nastaju ako se neuspješno uklone krive korespondencije. Proces triangulacije također može uzrokovati veliku grešku. Bilo kakve nesavršenosti u geometriji poput male kalibracijske greške će utjecati na trianguliranu poziciju 3D točaka što će na kraju utjecati na procjenu gibanja kamere.

4. Ispitni skupovi i programska implementacija

4.1. Ispitni skupovi

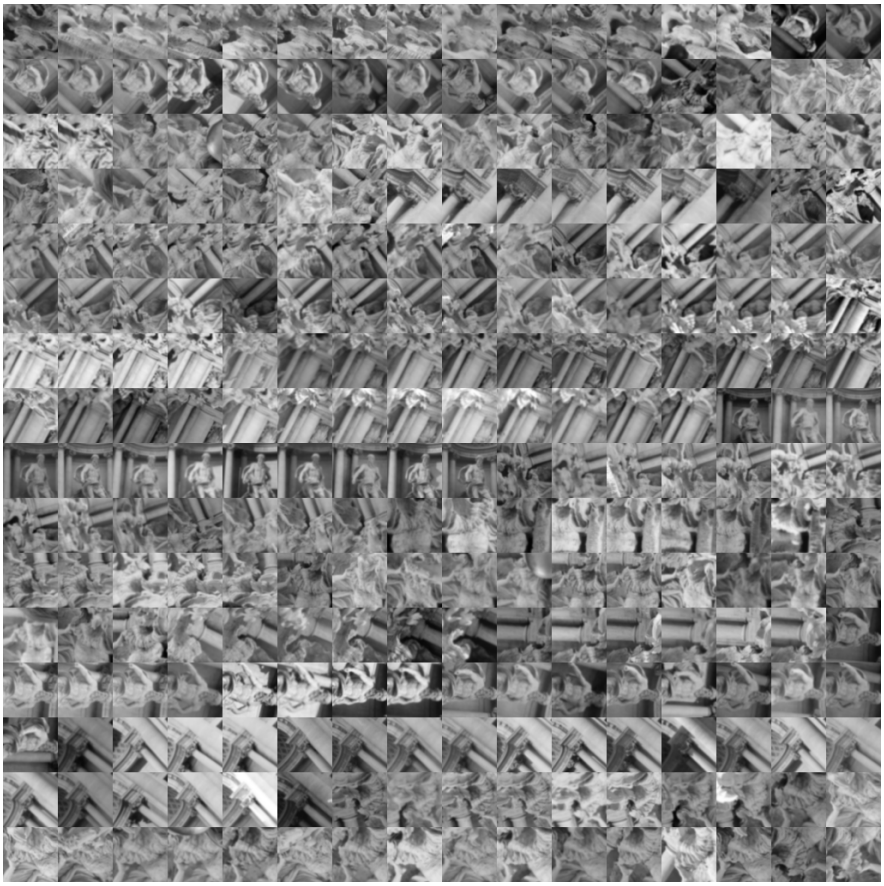
4.1.1. Ispitni skup KITTI

Ispitni skup KITTI [5] se sastoji od 22 stereo snimke snimljene u gradskoj i izvangradskoj vožnji sa stereo sustavom pričvršćenim na krov vozila. Slike s kamera su dohvaćane frekvencijom od 10Hz te spremljene u png formatu. Za 11 snimki su javno dostupni izvorni položaji kamera dobiveni s drugim senzorima te su se zbog tog razloga samo te snimke koristile za evaluaciju. Ovaj skup koristimo za evaluaciju estimacije gibanja.

Slika 4.1: Lijeva i desna slika ispitnog skupa KITTI

4.1.2. Ispitni skup za evaluaciju metrike

Za evaluaciju metrike se koristi ispitni skup predstavljen u [14]. U tom radu su predstavljena 3 skupa: Yosemite, Notre Dame i Liberty. U ovom radu se koristio ispitni skup Notre Dame. Skup se sastoji od niza slika dimenzija 1024×1024 u bmp formatu. Svaka slika se sastoji od 16×16 2D značajki opisanih s okvirom veličine 64×64 piksela. Za svaku takvu značajku se zna koju 3D točku predstavlja a da bi skup imao smisla ista 3D točka je prisutna na barem dvije 2D značajke. Primjer jedne slike iz takvog skupa se nalazi na slici 4.2.



Slika 4.2: Jedna slika iz Notre Dame ispitnog skupa

4.2. Programska izvedba

Programsko rješenje zadatka rada implementirano je u C++ programskom jeziku. Program je podijeljen na nekoliko modula te koristi nekoliko vanjskih biblioteka.

4.2.1. Vanjske biblioteke

OpenCV

Programska implementacija ovog rada oslanja se medu ostalim i na biblioteku OpenCV (engl. Open Source Computer Vision). Ova biblioteka otvorenog koda pruža niz metoda iz područja računalnog vida i obrade slika. Biblioteka je napisana u programskom jeziku C++, ali sadrži i sučelja za Python, C i Javu. Podržana je na većini bitnih operacijskih sustava. U radu se koriste različite funkcije implementirane u sklopu OpenCV-a.

Tensorflow

Tensorflow je biblioteka otvorenog koda za oblikovanje metoda strojnog učenja s naglaskom na sljedeće ključne mogućnosti:

- automatsko diferenciranje
- izvršavanje na raspodijeljenim i GPU arhitekturama
- bogato klijentsko sučelje prema Pythonu

Tensroflow također ima klijentsko sučelje prema C++-u, Javi i C-u. Tensorflow je zbog čiste arhitekture, bogate biblioteke, efikasne implementacije i sveobuhvatne dokumentacije trenutno najbolji izbor za istraživačke primjene a u ovom radu se koristi za učitavanja i izvođenje unaprijednih prolaza kroz konvolucijsku neuronsku mrežu.

Boost

Boost je biblioteka koja proširuje funkcionalnost standardne biblioteke programskog jezika C++. Sastoji se od skupa biblioteka opće namjene koje uključuju pametne pokazivače, apstrahiranje OS-a, operacije linearne algebre te ostalih biblioteka koje olakšavaju rad korisniku jezika C++. U ovom radu se boost koristi za rad s kvaternionima i implementaciju sferične linearne interpolacije

Libviso2

Libviso biblioteka [4] je jako brza, višeplatformska biblioteka za računanje vlastitog gibanja kamere napisana u C++. Stereo verzija se temelji na minimizaciji reprojekcijske pogreške rijetkih korespondencija značajki. Biblioteka ne pretpostavlja nikakva ograničenja na gibanje kamere ali zahtijeva da su ulazne slike rektificirane i da su poz-

nati kalibracijski parametri kamera. Ova biblioteka je korištena kao baza razvijenog sustava. Biblioteka je blago izmijenjena kako bi se omogućila lakša nadogradnja.

4.2.2. Moduli

Estimacija gibanja

Modul za estimaciju gibanja se nalazi u `src/egomotion` direktoriju. Ovaj modul je odgovoran za estimaciju gibanja na temelju izračunatih korespondencija. Estimacija gibanja je je implementirana u klasi `Egomotion`. Klas ima isto javno sučelje kao i ekvivalentne klase iz libviso biblioteke. Koristi se odvojena estimacija rotacije i translacije. Za rotaciju se koristi metoda 5 točaka iz biblioteke OpenCV:

```
   cv::Point2d pp = cv::Point2d(param.calib.cu, param.calib.cv);
   E = cv::findEssentialMat(points1, points2, param.calib.f, pp,
cv::RANSAC, 0.9999, 1.0, mask);
   recoverPose(E, points1, points2, R, t, param.calib.f, pp, mask);
```

Za određivanje translacije se koristi metoda tri točke iz libviso2 biblioteke koja je prilagođena za estimaciju translacije s fiksiranom rotacijom.

U ovom modulu se također nalaze i funkcije za sferičnu linearnu interpolaciju definirane u `slerp.h`

Ekstrakcija konvolucijskih deskriptora

Ekstrakcija konvolucijskih deskriptora se obavlja u modulu koji se nalazi u `src/conv` direktoriju. Tu se nalazi razred `ConvNetwork` u kojemu se učitava konvolucijska neuronska mreža s diska te koja ostvaruje jednostavno sučelje za ekstrakciju deskriptora obje slike. U pozadini se zapravo obavlja unaprijedni prolaz kroz mrežu. Struktura `ImageDescriptor` opisuje sve deskriptore slike i omogućuje jednostavno dohvaćanje deskriptora pojedinih značajki.

Određivanje korespondencija

Modul za određivanje korespondencija se nalazi u `src/matcher` direktoriju. Osnovni razred je `FeatureMatcher` koji nasljeđuje libviso `Matcher` razred te omogućuje odabir proizvoljnog detektora značajki. Razred `ConvMatcher` nasljeđuje `FeatureMatcher` te je u njemu implementirana naučena konvolucijska metrika.

Primjer koda koji određuje sličnost dvije značajke na koordinatama $(u1, v1)$ i $(u2, v2)$ je:

```
float *first = first_image->getFeature(u1, v1);
float *second = second_image->getFeature(u2, v2);
float similarity = cblas_sdot(64, first, 1, second, 1);
```

Apstraktni razred `Detector` opisuje sučelje koje koriste klase za određivanje korespondencija prilikom detekcije značajki.

Ostali moduli

U `src/libviso` se nalazi blago modificirana libviso2 biblioteka pogodna za nadogradnju. U `src/evaluation` se nalaze programi za evaluaciju metrika, a u `src/main` direktoriju se nalaze glavni programi. Ukupno su tri glavna programa u kojima je ujedno demonstrirano korištenje ostalih modula:

- `egomotion` - Osnovni program koji ne koristi naučenu metriku.
- `egomotion_slerp` - Program koji demonstrira korištenje sferične linearne interpolacije.
- `egomotion_conv` - Program koji koristi naučenu konvolucijsku metriku. Program je višedretveni kako bi se ubrzalo izvođenje. Jedna dretva računa deskriptore slika izvršavajući unaprijedne prolaze kroz neuronsku mrežu, dok druga dretva radi detekciju značajki i određivanje njihovih korespondencija.

4.3. Upute za instalaciju

Potrebno je imati instalirano:

- GCC 4.9.2+
- CMake 2.8+
- OpenCV 3.1
- Tensorflow 1.0 [1]
- OpenBLAS
- OpenMP

[1] Upute za instalaciju: `https://github.com/cjweeks/tensorflow-cmake`

Instalacija se obavlja izvršavanjem sljedećih naredbi:

```
git clone https://github.com/dario2332/viso
cd viso
mkdir build
cd build
cmake ..
make
```

5. Rezultati i evaluacija

U ovom poglavlju su predstavljeni svi rezultati ovoga rada. Prvo je prikazana evaluacija metrike i estimatora gibanja a na kraju je prikazano koliko metrika utječe na estimaciju gibanja.

5.1. Evaluacija metrike

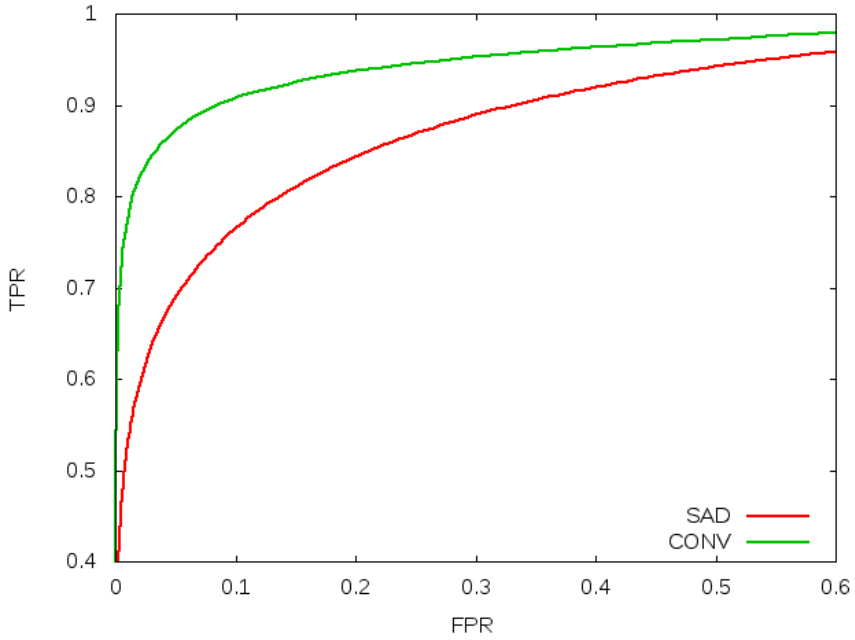
Evaluacija metrike se radila na ispitnom skupu opisanom u 4.1.2. Prvo su se odredili svi pozitivni parovi značajki a potom se nasumično generirao jednak broj negativnih parova značajki. Nisu promatrani svi parovi značajki jer bi onda bilo previše negativnih primjera. Par značajki se smatra pozitivnim ako obje značajke predstavljaju istu 3D točku. U suprotnome je par negativan.

Za svaku metriku se generira ROC (engl. Receiver operating characteristic) krivulja. Metriku kojom se određuju korespondencije značajki možemo promatrati kao binarni klasifikator tako da odredimo određeni prag. Ako je greška za neki specifični primjer ispod toga praga primjer smatramo pozitivnim (točna korespondencija). U suprotnome primjer smatramo negativnim (kriva korespondencija). Zatim se izračunaju sljedeća dva izraza na temelju predikcija metrike i pravih vrijednosti iz ispitnog skupa:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{5.1}$$

Gdje su TP (engl. true positive) broj pravih pozitivnih primjera, TN (engl. true negative) broj pravih negativnih primjera, FP (engl. false positive) broj krivih pozitivnih primjera i FN (engl. false negative) broj krivih negativnih primjera. TPR (engl. true pozitive rate) predstavlja postotak pravih pozitivnih primjera s obzirom na sve pozitivne primjere u skupu. Ovu vrijednost želimo maksimizirati. FPR (engl. false pozitive rate) predstavlja postotak krivih pozitivnih primjera s obzirom na sve negativne primjere u skupu. Ovu vrijednost želimo minimizirati.

ROC krivulja se generira tako da se ovaj postupak ponavlja više puta za različite vrijednosti praga klasifikacije te se onda dobivene vrijednosti iscrtaju na graf tako da



Slika 5.1: ROC krivulje konvolucijske i SAD metrike

je na x osi FPR a na y osi TPR. Što je više ROC krivulja bliža gornjem lijevom kutu (TPR = 1, FPR = 0) to je klasifikator bolji.

Na slici 5.1 možemo vidjeti ROC krivulje naučene konvolucijske korespondencijske metrike(označena s CONV) i SAD metrike na rijetkom skupu piksela u domeni derivacije slike koja se originalno koristi u libviso biblioteci. Ove metrike su opisane u poglavlju 2.3.

Na grafu se vidi da je konvolucijska metrika značajno bolja od SAD metrike koja se koristi u libviso biblioteci.

5.2. Estimacija gibanja

Evaluacija estimacije gibanja se radila na ispitnom skupu KITI. Za evaluaciju se koristio program dostupan na `http://kitti.is.tue.mpg.de/kitti/devkit\_odometry.zip`. U ovom potpoglavlju se analizira utjecaj odvojene estimacije rotacije metodom pet točaka na točnost estimacije gibanja. Također se analizira utjecaj sferične linearne interpolacije.

Rezultati evaluacije su prikazani u tablici 5.1. S t su označene pogreške translacije u postotcima a s R pogreške rotacije u stupnjevima po metru. Uspoređene su tri različite konfiguracije. U prvoj se koristila klasična biblioteka libviso2 bez dodatnih funkcionalnosti. U preostale dvije konfiguracije je korišteno sortiranje značajki po sta-

Sekvenca	Duljina[m]	libviso2		Metoda 5 točaka		SLERP	
		t(%)	R(°/m)	t(%)	R(°/m)	t(%)	R(°/m)
00	3724	2.324	0.0112	1.045	0.053	0.837	0.0036
01	2453	5.074	0.0079	7.373	0.0076	6.732	0.0044
02	5067	2.087	0.0078	1.258	0.0053	0.905	0.0036
03	561	2.376	0.0121	1.976	0.0074	1.477	0.0030
04	394	0.832	0.0074	1.576	0.0034	1.496	0.0023
05	2206	1.715	0.0097	0.947	0.0042	0.760	0.0028
06	1233	0.973	0.0054	1.124	0.0046	0.921	0.0025
07	695	1.859	0.0144	1.740	0.0080	1.327	0.0045
08	3223	2.349	0.0105	1.453	0.0052	1.036	0.0037
09	1705	2.335	0.0093	1.920	0.0056	1.293	0.0035
10	920	1.350	0.0084	1.205	0.0055	0.774	0.0030
Sve	22181	2.232	0.0095	1.550	0.0053	**1.208**	**0.0034**

Tablica 5.1: Rezultati na skupu KITTI

rosti. U drugoj konfiguraciji se koristila monokularna estimacija rotacije metodom pet točaka ali bez sferične linearne interpolacije dok je u trećoj konfiguraciji korištena i metoda sferične linearne interpolacije (engl. SLERP). Kao što vidimo monokularna estimacije metodom pet točaka značajno smanjuje prosječnu rotacijsku pogrešku a time ujedno i translacijsku pogrešku. Ako se koristi i sferična linearna interpolacija prosječne pogreške još dodatno padaju. Zadnje dvije konfiguracije su izvođene u isto vrijeme radi bolje mogućnosti usporedbe. To znači da se sferična linearna interpolacija računala s istim rotacijama koje su dobivene u drugoj konfiguraciji.

Najmanja prosječna pogreška je 1.208 % u translaciji i 0.0034 stupnjeva po metru u rotaciji. To je smanjenje za više od 1% u translaciji s obzirom na libviso2. Što se tiče rotacijske pogreške pogreška je skoro tri puta manje nego ona kod libviso2.

5.3. Konačni rezultati

U ovom poglavlju prikazujemo rezultate estimacije gibanja prilikom korištenja naučene konvolucijske metrike. Rezultati su prikazani u tablici 5.2. S libviso2+ je označena metoda odvojene estimacije rotacije i translacije bez sferične linearne interpolacije. Zatim je prikazana ista metoda s naučenom konvolucijskom metrikom. Na kraju su prikazani rezultati ako detektor iz libviso biblioteke zamijenimo FAST detektorom

Sekvenca	Duljina[m]	libviso2+		Konv. metrika		Konv. metrika + FAST	
		t(%)	R(°/m)	t(%)	R(°/m)	t(%)	R(°/m)
00	3724	1.045	0.053	1.190	0.0049	1.714	0.0069
01	2453	7.373	0.0076	5.834	0.0062	10.517	0.0304
02	5067	1.258	0.0053	1.186	0.0042	1.497	0.0068
03	561	1.976	0.0074	1.331	0.0052	2.309	0.0066
04	394	1.576	0.0034	1.933	0.0040	1.804	0.0037
05	2206	0.947	0.0042	1.309	0.0050	2.029	0.0089
06	1233	1.124	0.0046	1.199	0.0037	2.101	0.0044
07	695	1.740	0.0080	2.223	0.0110	1.522	0.0087
08	3223	1.453	0.0052	1.266	0.0061	1.972	0.0074
09	1705	1.920	0.0056	1.437	0.0058	2.085	0.0077
10	920	1.205	0.0055	1.115	0.0058	1.925	0.0119
Sve	22181	1.550	0.0053	**1.475**	**0.0052**	2.210	0.0085

Tablica 5.2: Konačni rezultati na skupu KITTI

iz OpenCV-a. Parametri su namješteni tako da je u svakoj metodi otprilike odabran jednak broj korespondencija. To znači da su se za libviso detektor uzimale samo dvije značajke po grupi, dok je za FAST detektor taj broj povišen na 5. Korespondencije se unutar svake grupe sortiraju prema starosti. Isprobano je sortiranje i po sličnosti značajki ali nije donijelo poboljšane rezultate.

Iz tablice vidimo da konvolucijska metrika ima mali utjecaj na translacijsku i rotacijsku pogrešku što upućuje na to da metrika nije ključan faktor u estimaciji gibanja. Zbog toga je metrika isprobana s "boljim" detektorom poput FASTA ali su rezultati estimacije gibanja postali značajno lošiji. Metrika je naknadno isprobana i s Harrisovim detektorom ali su rezultati još lošiji od FAST-a pa nisu ovdje prikazani.

Moguća objašnjenja lošijih rezultata s FAST detektorom su sljedeća:

1. Proučavajući ukupan broj odabranih značajki ustanovljeno je da je broj detektiranih značajki s FAST detektorom nešto ispod 10000 po slici dok je s libviso detektorom iznad 20000. Promjenom parametara je moguće povećavanje broja FAST značajki ali je ograničavajući faktor složenost. Određivanje korespondencija je složenosti $O(n^2)$. S takvom složenošću vrijeme izvođenja raste iznad 1s po paru slika ako broj značajki povećamo iznad 10000. Libviso detektor može podnijeti tako velik broj značajki zato što se značajke dijele na 4 skupine i određivanje korespondencija se obavlja samo unutar istih skupina.

2. S većim brojem značajki libviso detektora je ujedno ravnomjernije pokriveno i veće područje slike dok su kod FAST detektora značajke koncentriranije na specifičnim mjestima. Zato se prilikom grupiranja za libviso detektor uzima manji broj značajki po grupi da bi se dobio jednak ukupan broj korespondencija.

3. Dodatno, libviso detektor je detektor mrlja i kuteva dok je FAST samo detektor kuteva. Uz to, mrlje i kutevi se klasificiraju prema odzivu u maksimume i minimume. Vrlo je vjerojatno da će iste značajke na dvije različite slike biti svrstane u istu skupinu što je još jedan dodatni faktor koji omogućava bolje određivanje korespondencija.

6. Zaključak

U ovom radu je proučen utjecaj naučene konvolucijske metrike na određivanje vlastitog gibanja kamere. Metrika je evaluirana i uspoređena s ugrađenom metrikom iz libviso biblioteke pomoću ROC krivulja. Naučena konvolucijska metrika je značajno bolja od ugrađene libviso metrike. Iako su se zbog prethodne činjenice očekivalo znatno smanjenje pogreške ukoliko libviso metriku zamijenimo konvolucijskom u postupku estimacije gibanja, to se nije dogodilo. Dobivene pogreške su 1.48% u translaciji i 0.0052 °/m što je smanjenje pogreške od samo 0.07% u translaciji i 0.0001 °/m u rotaciji. Ova razlika nije značajna i može se zaključiti da kvaliteta metrike nema utjecaja na kvalitetu estimacije gibanja. Sljedeći pokušani eksperiment jest bio promjena detektora značajki čime su se dobile povećane pogreške. Iz analize iz prethodnog poglavlja se može zaključiti da su broj značajki, ravnomjerna rasprostranjenost značajki po cijeloj slici i raznovrsniji tipovi detektiranih značajki bitniji faktori od kvalitete detektora značajki.

Pošto naučena metrika ne daje poboljšanja prilikom vlastitog određivanja gibanja stereoskopske kamere ona se ne isplati koristiti u ovom postupku zbog velikog gubitka performansi. Za obradu jednog para slika potrebno jest obaviti unaprijedni prolaz kroz konvolucijsku neuronsku mrežu te prebaciti dobivene deskriptore u RAM. Ovo je najskuplji dio postupka te je usko grlo cijelog procesa. Na računalu na kojem se ovaj postupak testiran se unaprijedni prolaz obavljao na GPU-u. Računalo ima sljedeće komponente: Intel Core i7-6700HQ CPU, NVIDIA GeForce GTX 950M, 8GB RAM. Vrijeme izvođenja po paru slika je bilo oko 0.6s što je znatno sporije od metode koja ne koristi naučenu metriku.

Svi navedeni rezultati upućuju na to da ključni faktor u određivanju vlastitog gibanja kamere nisu ni kvaliteta metrike ni kvaliteta detektora. Mogući ključni faktor u određivanju vlastitog gibanja kamere jest kvaliteta kalibracije kamere i određivanja kalibracijskih parametara. Daljnji rad na ovom području bi se trebao usredotočiti na te parametre.

LITERATURA

[1] Motilal Agrawal, Kurt Konolige, i Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. U *Computer Vision–ECCV 2008*, stranice 102–115. Springer, 2008.

[2] Herbert Bay, Tinne Tuytelaars, i Luc Van Gool. Surf: Speeded up robust features. U *Computer vision–ECCV 2006*, stranice 404–417. Springer, 2006.

[3] Igor Cvišić i Ivan Petrović. Stereo odometry based on careful feature selection and tracking. U *Mobile Robots (ECMR), 2015 European Conference on*, stranice 1–6. IEEE, 2015.

[4] Andreas Geiger, Julius Ziegler, i Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. U *Intelligent Vehicles Symposium (IV)*, 2011.

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, i Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, stranica 0278364913491297, 2013.

[6] Chris Harris i Mike Stephens. A combined corner and edge detector. U *Alvey vision conference*, svezak 15, stranice 10–5244. Manchester, UK, 1988.

[7] Christopher G Harris i JM Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[9] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[10] David Nistér, Oleg Naroditsky, i James Bergen. Visual odometry. U *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, svezak 1, stranice I–652. IEEE, 2004.

[11] Edward Rosten i Tom Drummond. Machine learning for high-speed corner detection. U *Computer Vision–ECCV 2006*, stranice 430–443. Springer, 2006.

[12] Davide Scaramuzza i Friedrich Fraundorfer. Visual odometry [tutorial]. *Robotics & Automation Magazine, IEEE*, 18(4):80–92, 2011.

[13] Jianbo Shi i Carlo Tomasi. Good features to track. U *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, stranice 593–600. IEEE, 1994.

[14] Simon AJ Winder i Matthew Brown. Learning local image descriptors. U *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, stranice 1–8. IEEE, 2007.

[15] Jure Zbontar i Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17 (1-32):2, 2016.

Određivanje vlastitog gibanja kamere primjenom naučene korespondencijske metrike

Sažetak

Širi kontekst rada jest određivanje gibanja kamere analizom slijeda stereoskopskih slika. Većina pristupa za rješavanje tog problema temelji se na korespondenciji točkastih značajki u četvorkama slika pribavljenih u susjednim vremenskim trenutcima. U ovom radu se razmatra specifičan slučaj u kojemu se korespondencije ostvaruju ugrađivanjem slikovnih okana u visokodimenzionalni metrički prostor. U okviru rada su proučeni postupci određivanja korespondencija, praćenja značajki, te procjene gibanja. Razvijen je napredni sustav estimacije gibanja na temelju libviso biblioteke. U sustav je ugrađena metoda određivanja korespondencija značajki korištenjem naučene korespondencijske metrike. Preciznost postupaka je evaluirana na dva ispitna skupa. Provedena je evaluacija metrike a zatim evaluacija estimacije gibanja. Prikazani su i analizirani dobiveni rezultati te je predložen daljnji smjer razvoja.

Ključne riječi: određivanje vlastitog gibanja kamere, vizualna odometrija, estimacija gibanja, korespondencijska metrika, konvolucijska metrika, detekcija značajki, praćenje značajki, određivanje korespondencija, OpenCV, Libviso

Ego-motion estimation with a trained correspondence metric

Abstract

The broader context of this paper is camera ego-motion estimation through analysis of acquired stereo image sequences. Most approaches to solving this problem are based on feature point correspondences on two image pairs acquired at subsequent time intervals. In this paper, we consider a specific case in which correspondences are calculated by mapping image frames to high dimensional metric space. We studied different methods of feature matching, feature tracking, and ego-motion estimation. We developed an advanced system for ego-motion estimation based on libviso library. In this system, we incorporated new feature matching method based on trained correspondence metric. The precision of these methods is evaluated on two datasets. First, we evaluate trained correspondence metric and then ego-motion estimation is evaluated. Evaluation results are presented and analyzed and future improvements are proposed.

Keywords: ego-motion estimation, visual odometry, correspondence metric, convolutional metric, feature detection, feature matching, feature tracking, OpenCV, Libviso